Plaintext Awareness via Key Registration

Jonathan Herzog

CIS, TOC, CSAIL, MIT

Context of this work

- Originates from work on Dolev-Yao (DY) model
  - Symbolic approach to cryptography
  - From formal methods community
- In particular, previous work:
  1. Extracted a computational interpretation of Dolev-Yao assumptions, and
  2. Showed these assumptions to be satisfied by plaintext-aware (PA) encryption
- Led to interest in plaintext-aware (PA) encryption

Other results

- Thesis also contains direct extensions of DY work:
  - Strictly stronger interpretation of DY model
  - Proof that stronger interpretation satisfied by chosen-ciphertext security
  - Computationally sound extensions (Diffie-Hellman)

Overview

- This talk: self-contained work on plaintext awareness
- Strongest known security definition for public-key encryption
- However, current definition is problematic
- This work: removing problems in definition, keeping strength

[With Moses Liskov and Silvio Micali, CRYPTO 2003]

Plaintext awareness

- A public-key encryption scheme consists of
  - G: key-generation algorithm
  - E: encryption algorithm, and
  - D: decryption algorithm
- An encryption scheme is PA if
  1. It keeps the plaintext secret, and
  2. Adversary “knows” plaintext to any ciphertext it creates
- But what do we actually mean?

Secrecy

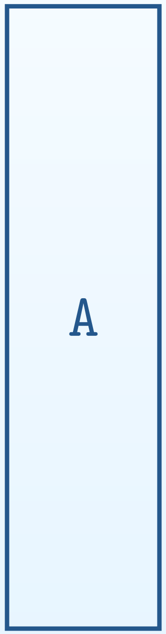
- Weakest standard definition of secrecy is *semantic security* [GM84]:

  "No adversary can do better than random in when trying to distinguish encryptions of m_0 from encryptions of m_1 (under a randomly chosen key) even if it gets to pick m_0 and m_1 itself."

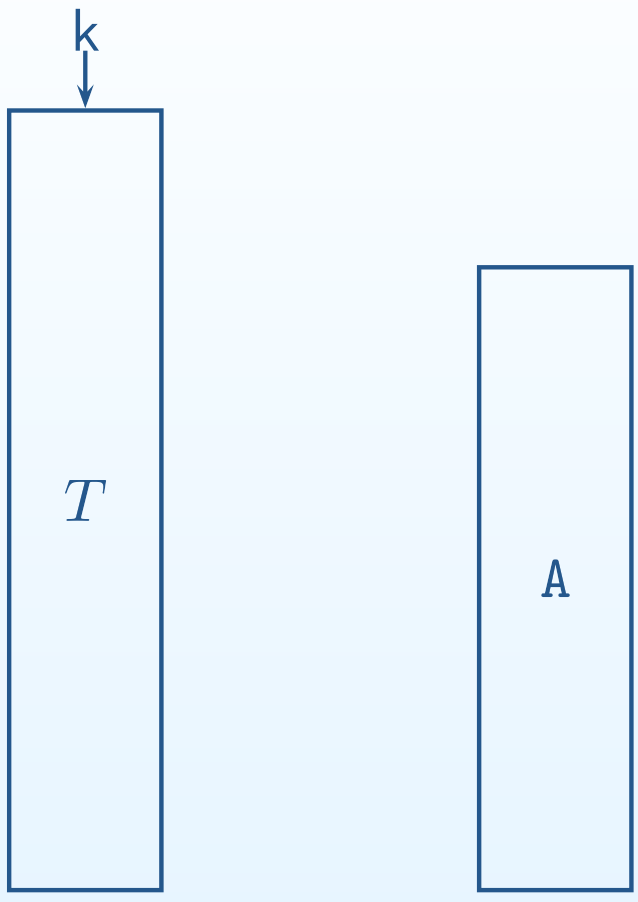
- Show same formalization twice: graphically and in standard (GMR) notation

Semantic security

$\forall \mathtt{A}_{PPT}$

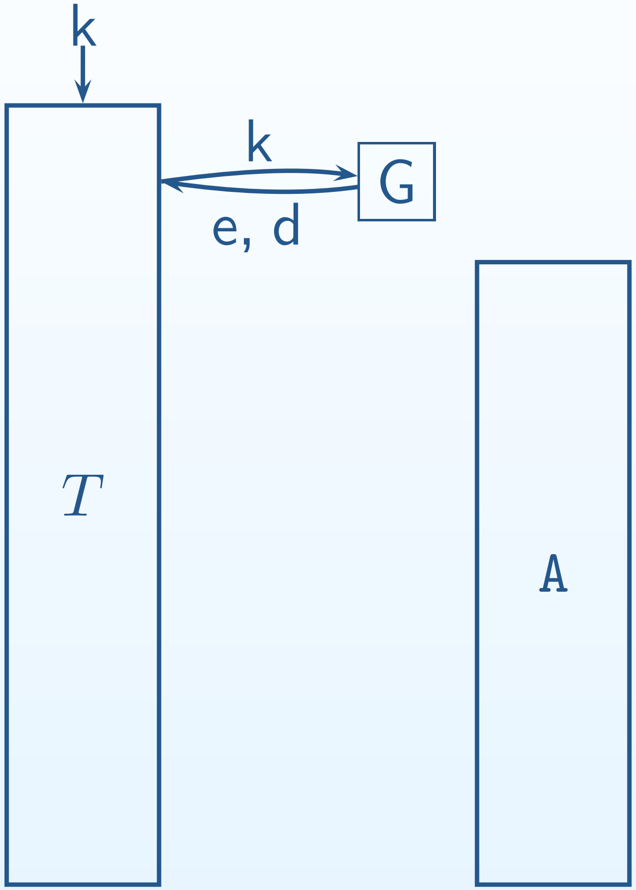


Semantic security



$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

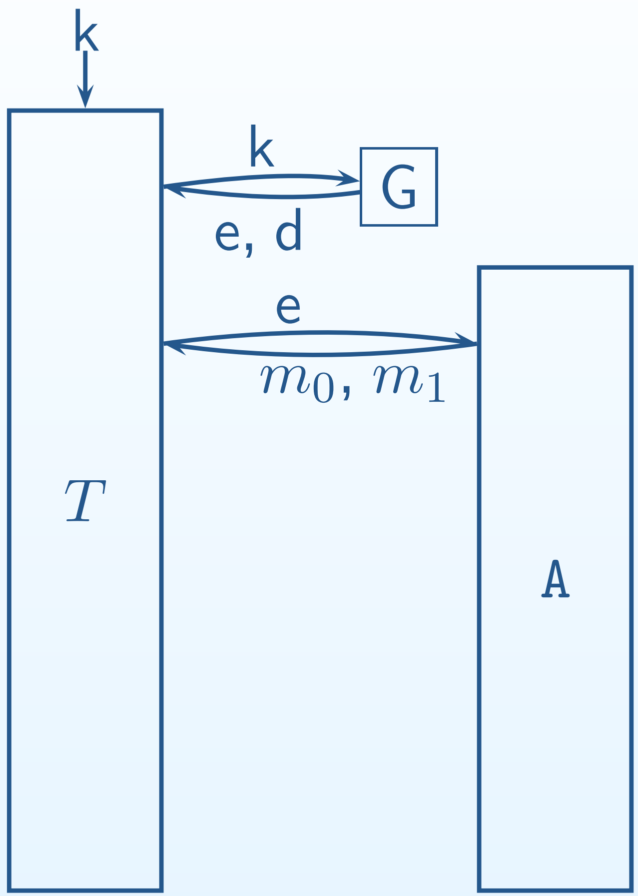
Semantic security



$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

$$(\mathsf{e}, \mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$

Semantic security

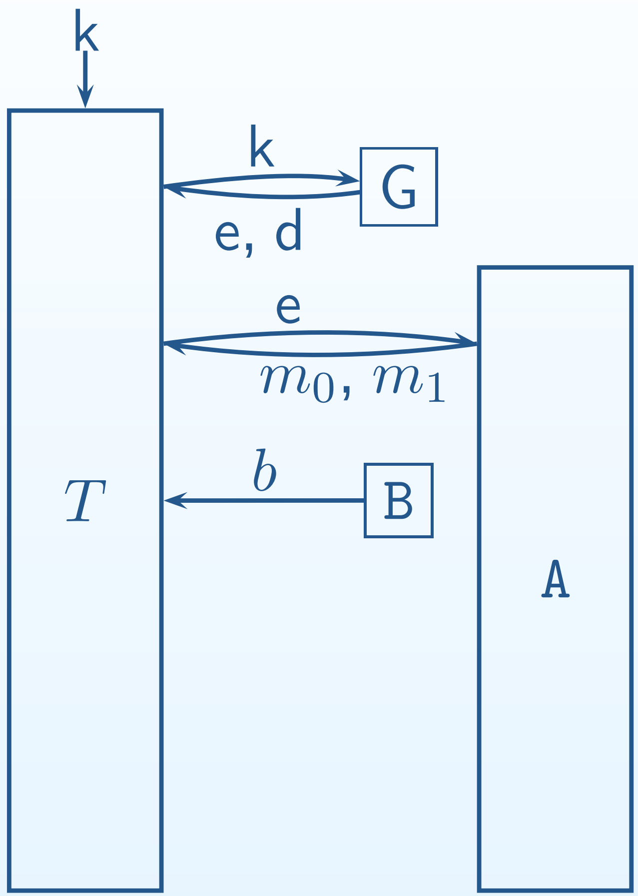


$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

$$(\mathsf{e}, \mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$

$$m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$

Semantic security



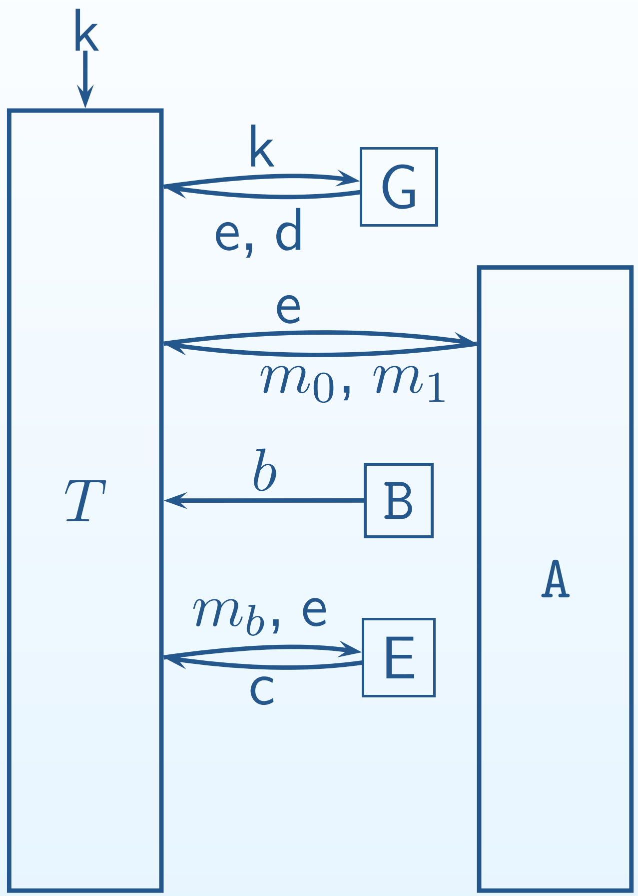
$$\forall \mathtt{A}_{PPT}\ \forall \text{ s. l. } \mathsf{k}$$

$$(\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$

$$m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$

$$b \leftarrow \mathtt{CoinFlip}(0,1);$$

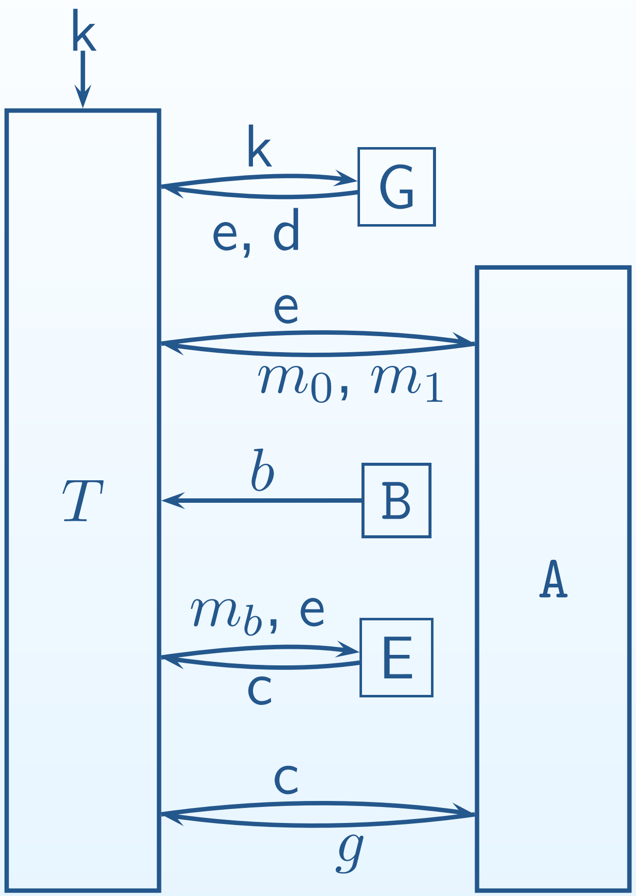
Semantic security



$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

$$(\mathsf{e}, \mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$
$$b \leftarrow \mathtt{CoinFlip}(0, 1);$$
$$\mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e});$$

Semantic security

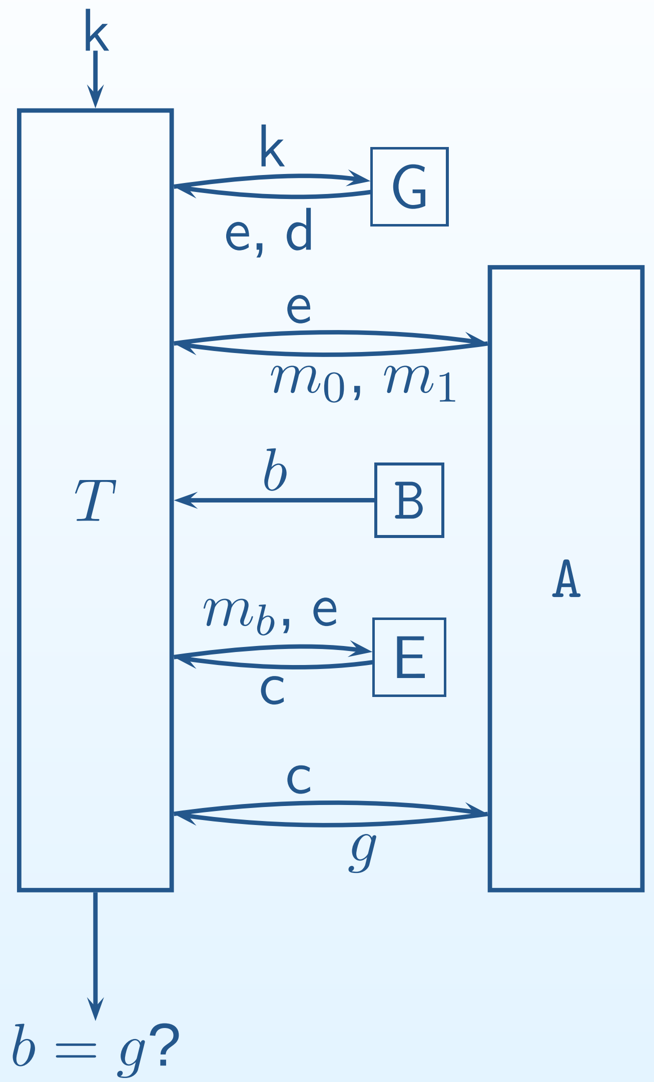


$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

$$(\mathsf{e}, \mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$
$$b \leftarrow \mathtt{CoinFlip}(0, 1);$$
$$\mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e});$$
$$g \leftarrow \mathtt{A}(\mathsf{c}) :$$

($\mathtt{A}$ keeps state)

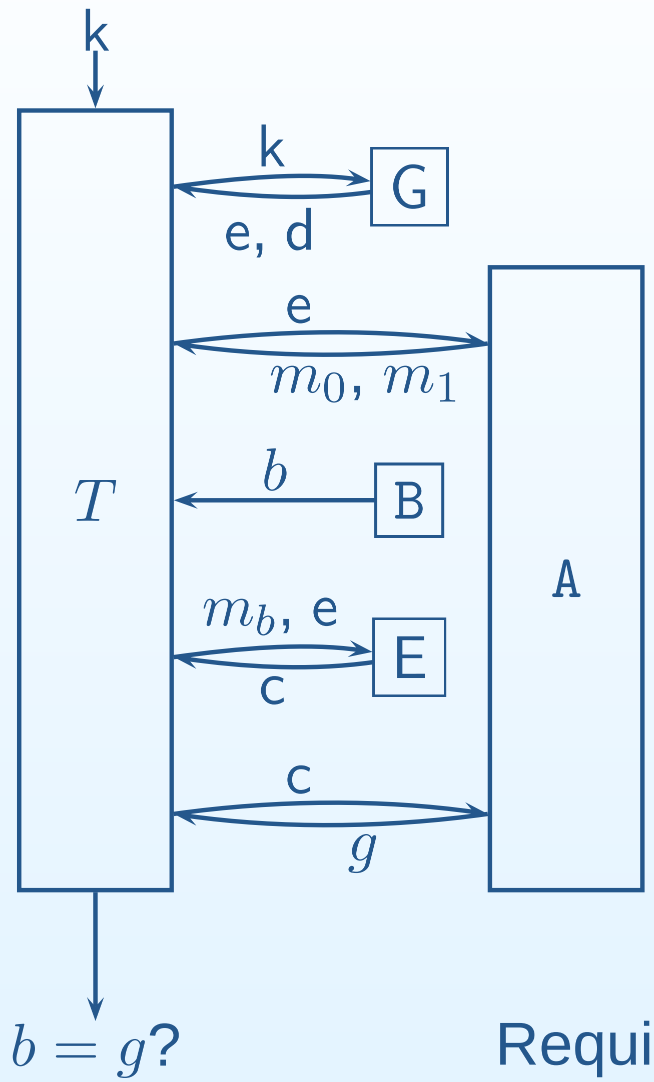
Semantic security



$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

$$(\mathsf{e}, \mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$
$$b \leftarrow \mathtt{CoinFlip}(0, 1);$$
$$\mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e});$$
$$g \leftarrow \mathtt{A}(\mathsf{c}) :$$
$$b = g$$

Semantic security



$\forall \mathtt{A}_{PPT}\ \forall$ s. l. k

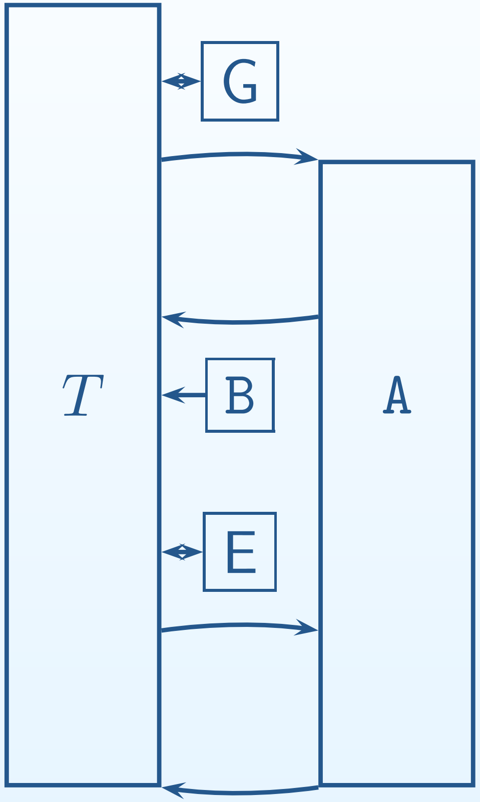
$$
\begin{aligned}
\Pr[\quad & (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
& m_0, m_1 \leftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e}); \\
& b \leftarrow \mathtt{CoinFlip}(0,1); \\
& \mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e}); \\
& g \leftarrow \mathtt{A}(\mathsf{c}) : \\
& b = g] \leq \tfrac{1}{2} + neg(\mathsf{k})
\end{aligned}
$$

Require: $\Pr[b = g] \cong \frac{1}{2}$

Strengthening semantic security

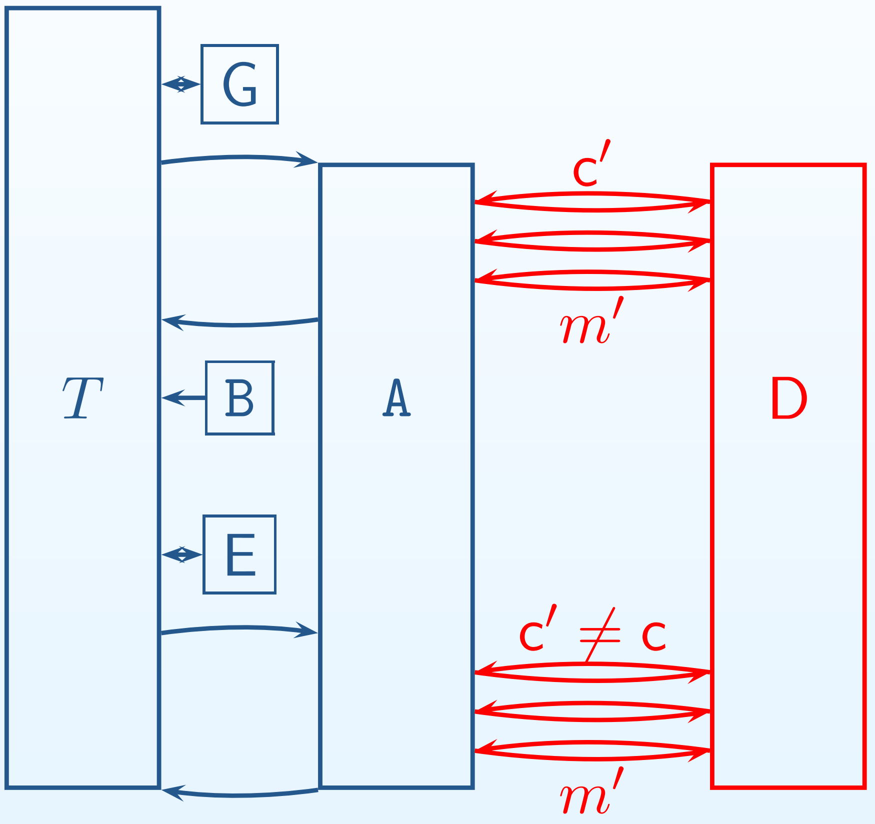
- Semantic security not strong enough for many applications
  - Cannot be used in protocols
  - Honest participants might provide to adversary services not captured by definition
- Two ways to strengthen:
  1. Chosen-ciphertext attack
  2. Plaintext awareness

Security against the chosen-ciphertext attack



$$
\begin{aligned}
&\forall \mathtt{A}_{PPT}\\
&\Pr[\quad (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});\\
&\qquad m_0, m_1 \leftarrow \mathtt{A} \qquad (1^{\mathsf{k}}, \mathsf{e});\\
&\qquad b \leftarrow \mathtt{CoinFlip}(0,1);\\
&\qquad \mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e});\\
&\qquad g \leftarrow \mathtt{A} \qquad (\mathsf{c}) :\\
&\qquad b = g] \leq \frac{1}{2} + neg(\mathsf{k})
\end{aligned}
$$

Security against the chosen-ciphertext attack



$$\forall \mathtt{A}_{PPT}$$

$$\begin{aligned}\Pr[\quad & (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});\\ & m_0, m_1 \leftarrow \mathtt{A}^{\mathsf{D}(\cdot,\mathsf{d})}(1^{\mathsf{k}},\mathsf{e});\\ & b \leftarrow \mathtt{CoinFlip}(0,1);\\ & \mathsf{c} \leftarrow \mathsf{E}(m_b,\mathsf{e});\\ & g \leftarrow \mathtt{A}^{\mathsf{D}(\cdot\neq\mathsf{c},\mathsf{d})}(\mathsf{c}) :\\ & b = g] \leq \tfrac{1}{2} + neg(\mathsf{k})\end{aligned}$$

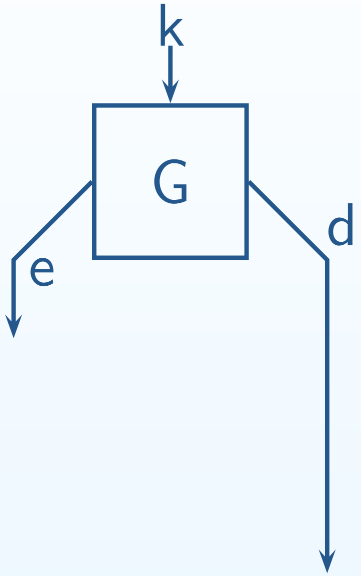
Plaintext awareness

- Another notion: plaintext awareness
- Intuition: adversary “knows” plaintext to any ciphertext it creates
- Algorithm “knowledge” is what can be calculated
- Adversary $\mathtt{A}$ knows x if $\mathtt{A}$ + another algorithm (called *extractor*) can compute x
- Plaintext awareness: there exists an extractor that can produce the plaintext to adversary’s ciphertext

Plaintext awareness

$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

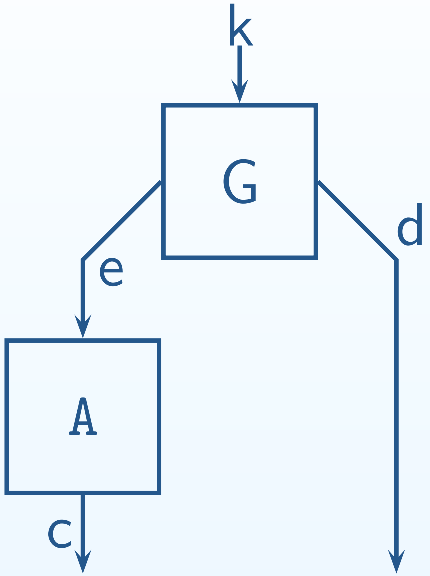
Plaintext awareness



$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

$$(\mathsf{e}, \mathsf{d}) \longleftarrow \mathsf{G}(1^{\mathsf{k}});$$

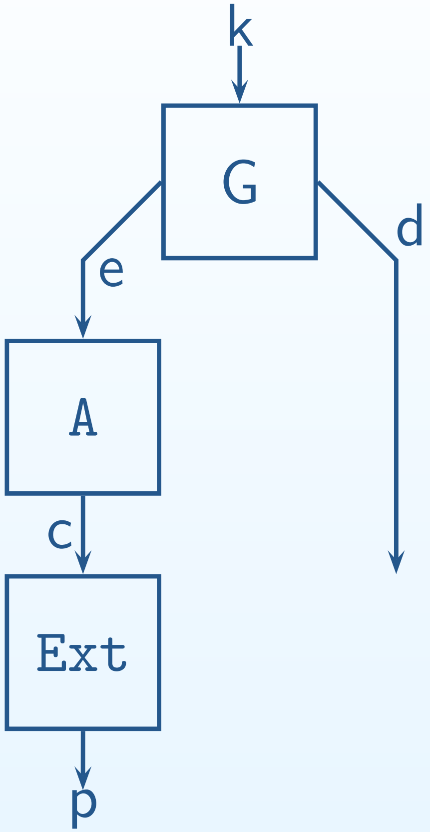
Plaintext awareness



$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$

$$(\mathsf{e}, \mathsf{d}) \longleftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$\mathsf{c} \longleftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$$

Plaintext awareness



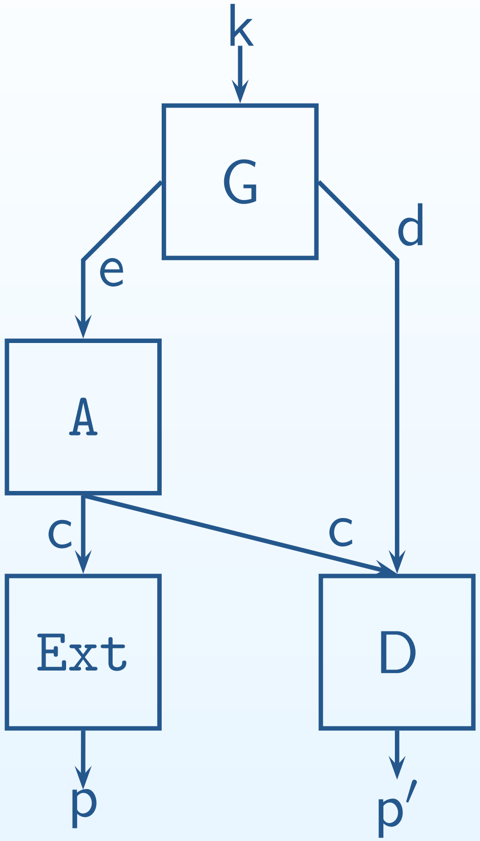
$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$

$(\mathsf{e}, \mathsf{d}) \longleftarrow \mathsf{G}(1^{\mathsf{k}});$

$\mathsf{c} \longleftarrow \mathtt{A}(1^{\mathsf{k}}, \mathsf{e});$

$\mathsf{p} \longleftarrow \mathtt{Ext}(1^{\mathsf{k}}, \mathsf{e}, \mathsf{c});$

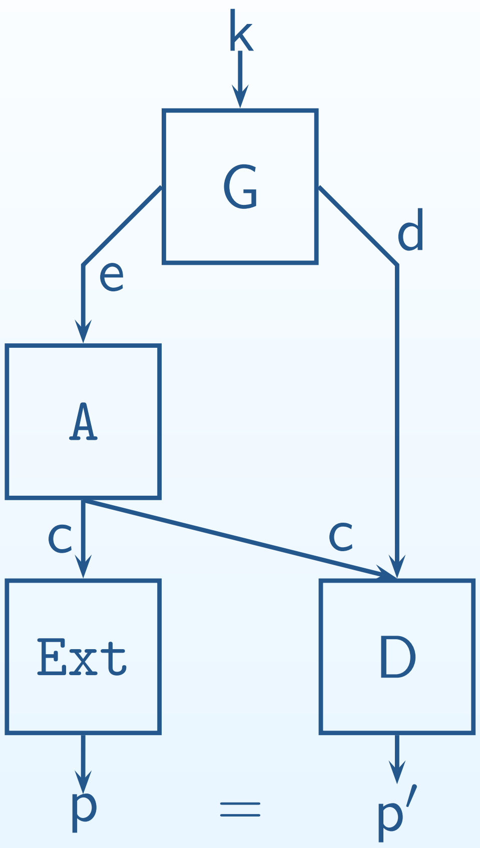
Plaintext awareness



$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$

$$(\mathsf{e},\mathsf{d}) \longleftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$\mathsf{c} \longleftarrow \mathtt{A}(1^{\mathsf{k}},\mathsf{e});$$
$$\mathsf{p} \longleftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c});$$
$$\mathsf{p}' \longleftarrow \mathsf{D}(\mathsf{c},\mathsf{d});$$

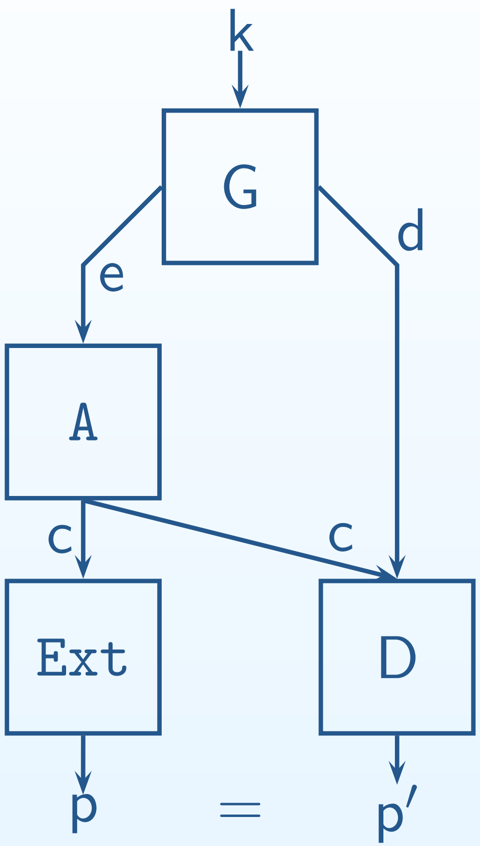
Plaintext awareness



$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

$$\Pr[\quad (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});$$
$$\mathsf{c} \leftarrow \mathtt{A}(1^{\mathsf{k}},\mathsf{e});$$
$$\mathsf{p} \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c});$$
$$\mathsf{p}' \leftarrow \mathsf{D}(\mathsf{c},\mathsf{d});$$
$$\mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k})$$

Plaintext awareness



$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

$$\begin{aligned} \Pr[\quad & (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\ & \mathsf{c} \leftarrow \mathtt{A}(1^{\mathsf{k}},\mathsf{e}); \\ & \mathsf{p} \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c}); \\ & \mathsf{p}' \leftarrow \mathsf{D}(\mathsf{c},\mathsf{d}); \\ & \mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k}) \end{aligned}$$

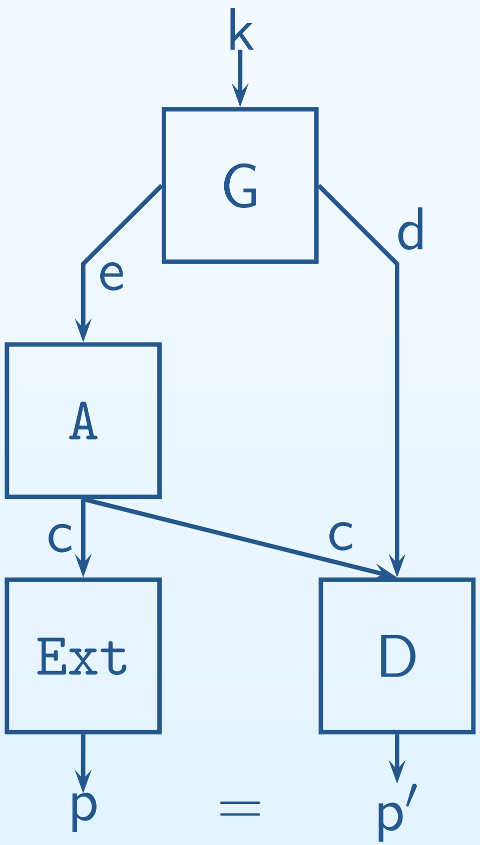
Do we want this?

Extractor requirements

- Note: extractor makes decryption oracle redundant
- Also violates semantic security
  - Takes in ciphertext, produces plaintext
- Solution: limit extractor to adversary's ciphertexts
- Make extractor use additional information from adversary
  - Ensure same information not available from honest participants

- Existing definition uses *random oracle*:
  - Oracle `O` that provides random function

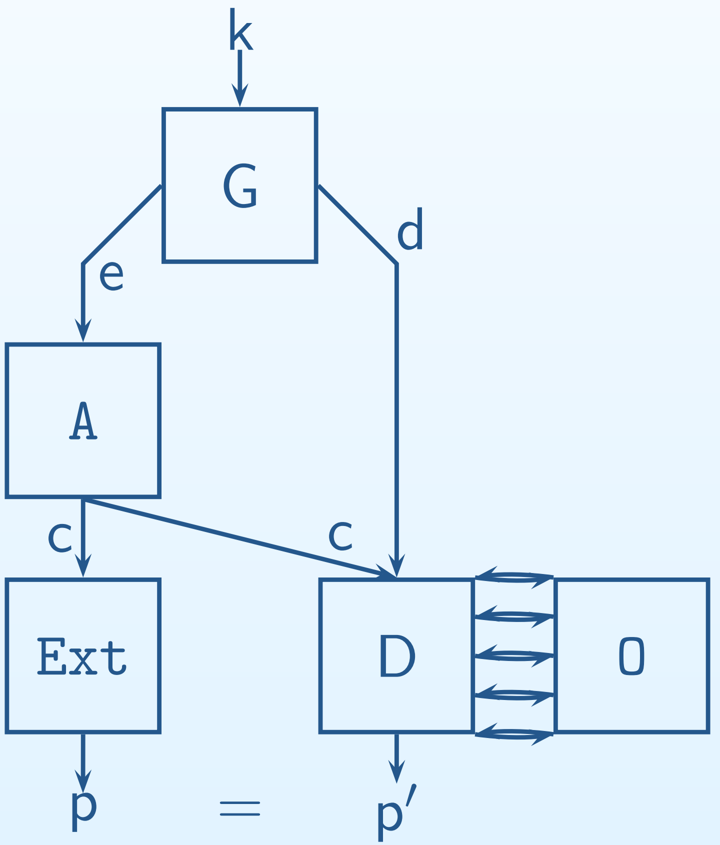
Plaintext awareness and the random oracle



$$
\begin{aligned}
&\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT} \\
&\Pr[\quad (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
&\qquad \mathsf{c} \leftarrow \mathtt{A} \quad (1^{\mathsf{k}},\mathsf{e}); \\
&\qquad \mathsf{p}' \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c},\quad); \\
&\qquad \mathsf{p} \leftarrow \mathsf{D} \quad (\mathsf{c},\mathsf{d}); \\
&\qquad \mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k})
\end{aligned}
$$

Plaintext awareness and the random oracle

- Encryption, decryption now use oracle

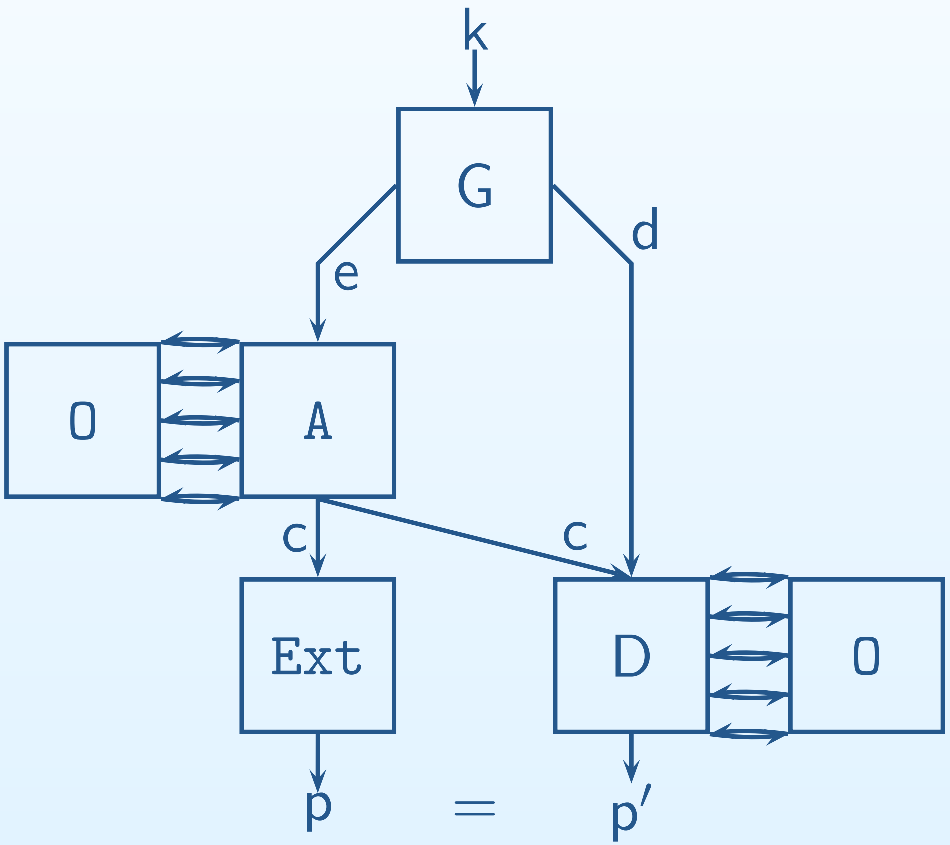


$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

$$\begin{aligned}\Pr[\quad & (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});\\ & \mathsf{c} \leftarrow \mathtt{A}\quad (1^{\mathsf{k}},\mathsf{e});\\ & \mathsf{p}' \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c},\quad);\\ & \mathsf{p} \leftarrow \mathsf{D}^{\mathtt{O}(\cdot)}(\mathsf{c},\mathsf{d});\\ & \mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k})\end{aligned}$$

Plaintext awareness and the random oracle

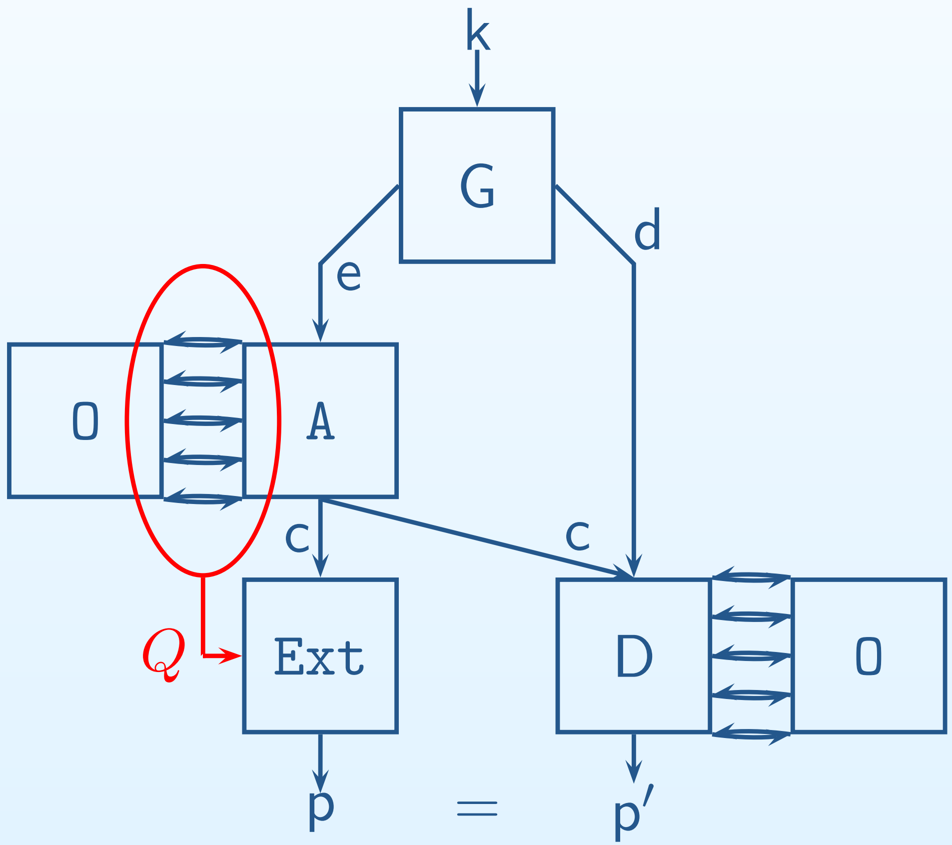
- Encryption, decryption now use oracle
- Adversary gets access to oracle



$$
\begin{array}{l}
\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT} \\
\Pr[\quad (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
\qquad \mathsf{c} \leftarrow \mathtt{A}^{\mathtt{O}(\cdot)}(1^{\mathsf{k}},\mathsf{e}); \\
\qquad \mathsf{p}' \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c},\quad); \\
\qquad \mathsf{p} \leftarrow \mathsf{D}^{\mathtt{O}(\cdot)}(\mathsf{c},\mathsf{d}); \\
\qquad \mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k})
\end{array}
$$

Plaintext awareness and the random oracle

- Encryption, decryption now use oracle
- Adversary gets access to oracle
- Extractor given oracle queries made by adversary



$$\exists \mathtt{Ext}\ \forall \mathtt{A}_{PPT}$$

$$\begin{aligned}\Pr[\quad & (\mathsf{e},\mathsf{d}) \leftarrow \mathsf{G}(1^{\mathsf{k}});\\ & \mathsf{c} \leftarrow \mathtt{A}^{\mathtt{O}(\cdot)}(1^{\mathsf{k}},\mathsf{e});\\ & \mathsf{p}' \leftarrow \mathtt{Ext}(1^{\mathsf{k}},\mathsf{e},\mathsf{c},Q);\\ & \mathsf{p} \leftarrow \mathsf{D}^{\mathtt{O}(\cdot)}(\mathsf{c},\mathsf{d});\\ & \mathsf{p} = \mathsf{p}'] \geq 1 - neg(\mathsf{k})\end{aligned}$$

Previous work

- This is original definition of PA [BR95]
  - Current definition is slight refinement [BDPR98]
  - Encryption scheme might be used in protocol
  - Provides adversary with source of externally-generated ciphertexts
    - Adversary wants to create ciphertext with unknown plaintext
    - Source of such ciphertexts might help
    - “Challenge” ciphertext must be new
- Known: PA $\subsetneq$ CC-security [ibid]
  - (CC-security still strongest possible without trusted third party)
- However, PA considered suspect, not widely used
  - Due to use of random oracle

Necessity of the Random Oracle

- Sometimes possible to replace oracle with algorithm
  - Abstraction of MD5, SHA-1
- Not possible in general [CGH98,GT03]
- Not possible in this case
- Interface with oracle gives extractor a “window” into adversary’s state
- Lost if oracle replaced with internal algorithm

Objections to the random oracle

Objections to the random oracle

1. Network overhead
   - E, D now use oracle also
   - Communication required for every operation

Objections to the random oracle

1. Network overhead
   - E, D now use oracle also
   - Communication required for every operation
2. Single *global* point of failure
   - Security depends on secrecy of queries
     - If the adversary gets queries, can run extractor to produce plaintext
   - Random oracle knows every message
     - Pray it's not corrupted!

Original Definition (concluded)

Original definition of PA

- Used extended model of public-key encryption
- Added unrealistic third party (oracle)
  - Required communication with oracle for every encryption/decryption
  - Trusts oracle with every message
- Alternately, dubious replacement
  - MD5 $\neq$ random oracle

Our Contribution

- Remove random oracle from PA
- Propose a more *natural* change to the model
  - Add a third party *already used in practice*
- Use that party only once
  - At key generation
- Trust that party with as little as possible
  - “Fail-safes” to CC-security when party corrupted
- Also show an general-assumption implementation

Status

- Previous definition
- **Our model**
- Our definition
- Our implementation

Our model

- Two kinds of key-pairs:
  - Receiver (e_r, d_r)
  - Sender (e_s, d_s)
    - "Sender" keys $\approx$ signature keys
- Encryption, decryption require both public keys
  - Encryption requires sender's private key
  - Decryption requires receiver's private key
- Public sending key registered with *Registration Authority* (RA)

Registration Authority

- Plays same role as certification authority
- Validates, publishes new public keys.
- Sender key generation and registration represented by protocol

$$\mathsf{User} \longleftrightarrow \mathsf{RA}$$

  - User outputs public, private keys
  - RA outputs (publishes) public key only
- Can think of RA issuing certificate for public key
- Implicitly assuming public-key infrastructure (PKI)
  - Bind key to names, vice-versa
  - Note: sender needs binding also
- For our purposes: RA validation ensures sender key has extractor

Status

- Previous definition
- Our model
- **Our definition**
- Our implementation

A two-part definition

- A scheme is *plaintext-aware via key registration* if:
  1. Honest RA $\Rightarrow$ plaintext awareness
     - “There exists an extractor such that, if the adversary creates a ciphertext relative to a *registered key*, then the extractor can re-create the plaintext”
     - As before, extractor needs additional information
     - Our definition: history of adversary’s internal state
  2. Chosen-ciphertext security
     - Even if RA is corrupt
     - (Best possible without trusted third party)

Chosen-ciphertext security

CC-security even if RA is corrupted:

$$\forall \text{ oracle-calling adversaries } \mathtt{A}$$

$$\begin{aligned}\Pr[\quad & (\mathsf{d}_r, \mathsf{e}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\ \\ & m_0, m_1 \leftarrow \mathtt{A}^{\mathsf{D}(\cdot, \mathsf{d}_r, \cdot)}(\mathsf{e}_r, \mathsf{e}_s); \\ & b \leftarrow \{0,1\}; \\ & \mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e}_r, \mathsf{d}_s); \\ & g \leftarrow \mathtt{A}^{\mathsf{D}(\cdot \neq \mathsf{c}, \mathsf{d}_r, \cdot)}(c) : \\ & b = g] \leq \tfrac{1}{2} + neg(\mathsf{k})\end{aligned}$$

Chosen-ciphertext security

CC-security even if RA is corrupted:

$\forall$ oracle-calling adversaries $\mathtt{A}$

$$
\begin{aligned}
\Pr[\quad & (\mathsf{d}_r, \mathsf{e}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
& (\mathsf{e}_s, \mathsf{d}_s) \xleftarrow{\mathsf{User}} (\mathsf{User} \leftrightarrow \mathtt{A})\ ; \\
& m_0, m_1 \leftarrow \mathtt{A}^{\mathsf{D}(\cdot, \mathsf{d}_r, \cdot)}(\mathsf{e}_r, \mathsf{e}_s); \\
& b \leftarrow \{0, 1\}\ ; \\
& \mathsf{c} \leftarrow \mathsf{E}(m_b, \mathsf{e}_r, \mathsf{d}_s); \\
& g \leftarrow \mathtt{A}^{\mathsf{D}(\cdot \neq \mathsf{c}, \mathsf{d}_r, \cdot)}(c)\ : \\
& b = g] \leq \tfrac{1}{2} + neg(\mathsf{k})
\end{aligned}
$$

Plaintext Awareness (1)

"There exists an extractor such that if the adversary creates a ciphertext with a registered key, then the extractor can re-create the plaintext."

- Who registers the key? User or adversary?
- Above should hold on both cases

$\forall$ adversaries $\mathtt{A}$, $\exists$ efficient algorithm $\mathtt{Ext}_{\mathtt{X}}$

$$
\begin{aligned}
Pr[\quad & (\mathsf{e}_r, \mathsf{d}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
& c \leftarrow \mathtt{A}(\mathsf{e}_{\mathtt{X}}, \mathsf{e}_r) : \\
& p \leftarrow \mathtt{Ext}_{\mathtt{X}}(c, \mathsf{e}_r, \mathsf{e}_{\mathtt{X}}); \\
& p' \leftarrow \mathsf{D}(c, \mathsf{d}_r, \mathsf{e}_{\mathtt{X}}) : \\
& p = p'] \geq 1 - neg(\mathsf{k})
\end{aligned}
$$

Plaintext Awareness (1)

"There exists an extractor such that if the adversary creates a ciphertext with a registered key, then the extractor can re-create the plaintext."

- Who registers the key? User or adversary?
- Above should hold on both cases

$\forall$ adversaries $\mathtt{A}$, $\forall\, \mathtt{X} \in \{\mathtt{A}, \mathsf{User}\}$ $\exists$ efficient algorithm $\mathtt{Ext}_{\mathtt{X}}$

$$\begin{aligned} Pr[\quad & (\mathsf{e}_r, \mathsf{d}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\ & (\mathsf{e}_{\mathtt{X}}, \mathsf{d}_{\mathtt{X}}) \xleftarrow{\mathtt{X}} (\mathtt{X} \leftrightarrow \mathsf{RA})\,; \\ & c \leftarrow \mathtt{A}(\mathsf{e}_{\mathtt{X}}, \mathsf{e}_r) : \\ & p \leftarrow \mathtt{Ext}_{\mathtt{X}}(c, \mathsf{e}_r, \mathsf{e}_{\mathtt{X}}); \\ & p' \leftarrow \mathsf{D}(c, \mathsf{d}_r, \mathsf{e}_{\mathtt{X}}) : \\ & p = p'] \geq 1 - neg(\mathsf{k}) \end{aligned}$$

Plaintext awareness (2)

- As in standard definition, extractor needs more than ciphertext
- We use internal *history* of adversary
  - Contains all inputs, randomness, state transitions
  - (Explicitly excluding erasure)

$$\forall \text{ adversaries } \mathtt{A}, \forall\, \mathtt{X} \in \{\mathtt{A}, \mathsf{User}\},$$
$$\exists \text{ efficient algorithm } \mathtt{Ext}_{\mathtt{X}}$$
$$\begin{aligned} Pr[\quad & (\mathsf{e}_r, \mathsf{d}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\ & (\mathsf{e}_{\mathtt{X}}, \mathsf{d}_{\mathtt{X}}) \leftarrow \mathsf{OUT}_{\mathtt{X},\mathsf{RA}}(\mathtt{X})\, \mathsf{e}_r, \cdot 1^{\mathsf{k}}, \cdot; \\ & \\ & c \leftarrow \mathtt{A}(\mathsf{e}_{\mathtt{X}}, \mathsf{e}_r) : \\ & \mathtt{Ext}_{\mathtt{X}}(\quad, c, \mathsf{e}_r, \mathsf{e}_{\mathtt{X}}) = \mathsf{D}(c, \mathsf{d}_r, \mathsf{e}_{\mathtt{X}}) \qquad] \geq 1 - neg(\mathsf{k}) \end{aligned}$$

Plaintext awareness (2)

- As in standard definition, extractor needs more than ciphertext
- We use internal *history* of adversary
  - Contains all inputs, randomness, state transitions
  - (Explicitly excluding erasure)

$\forall$ adversaries $\mathtt{A}, \forall\, \mathtt{X} \in \{\mathtt{A}, \mathsf{User}\}$,

$\exists$ efficient algorithm $\mathtt{Ext}_{\mathtt{X}}$

$$
\begin{aligned}
Pr[\quad & (\mathsf{e}_r, \mathsf{d}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
& (\mathsf{e}_{\mathtt{X}}, \mathsf{d}_{\mathtt{X}}) \leftarrow \mathsf{OUT}_{\mathtt{X},\mathsf{RA}}\,(\mathtt{X})\; \mathsf{e}_r, \cdot 1^{\mathsf{k}}, \cdot; \\
& h \xleftarrow{H} \mathtt{A}; \\
& c \leftarrow \mathtt{A}(\mathsf{e}_{\mathtt{X}}, \mathsf{e}_r) : \\
& \mathtt{Ext}_{\mathtt{X}}(h, c, \mathsf{e}_r, \mathsf{e}_{\mathtt{X}}) = \mathsf{D}(c, \mathsf{d}_r, \mathsf{e}_{\mathtt{X}}) \qquad] \geq 1 - neg(\mathsf{k})
\end{aligned}
$$

Plaintext awareness (3)

- Might as well allow adversary access to decryption oracle
- Encryption oracle?
  - Now necessary: encryption uses private keys
  - However, not general enough
- As before, adversary might be in protocol
  - Access to externally-generated ciphertexts
- Represent this as arbitrary *ally* oracle `L`
  - Looks at history of adversary
  - Performs some computation, produces plaintext
  - Encrypted, ciphertext given to adversary
- Encryption oracle functionality as special case
- Adversary's "challenge" ciphertext not from ally

Plaintext Awareness (4)

$\forall$ adversaries $\mathtt{A}, \forall\, \mathtt{X} \in \{\mathtt{A}, \mathsf{User}\}$,

$\exists$ efficient algorithm $\mathtt{Ext}_{\mathtt{X}}$, $\forall$ PPT allies $\mathtt{L}$,

$$
\begin{aligned}
Pr[\quad & (\mathsf{e}_r, \mathsf{d}_r) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\
& (\mathsf{e}_{\mathtt{X}}, \mathsf{d}_{\mathtt{X}}) \leftarrow \mathsf{OUT}_{\mathtt{X},\mathsf{RA}}\,(\mathtt{X})\; \mathsf{e}_r, \cdot 1^{\mathsf{k}}, \cdot; \\
& h \xleftarrow{H} \mathtt{A}; \\
& c \leftarrow \mathtt{A}^{\mathtt{L}'_{\mathsf{d}_{\mathtt{X}}}(\cdot), \mathsf{D}(\cdot, \mathsf{d}_r, \cdot)}(\mathsf{e}_{\mathtt{X}}, \mathsf{e}_r) : \\
& \mathtt{Ext}_{\mathtt{X}}(h, c, \mathsf{e}_r, \mathsf{e}_{\mathtt{X}}) = \mathsf{D}(c, \mathsf{d}_r, \mathsf{e}_{\mathtt{X}}) \text{ given that } c \text{ not from } \mathtt{L} \\
&] \geq 1 - neg(\mathsf{k})
\end{aligned}
$$

Status

- Previous definition
- Our model
- Our definition
- **Our implementation**

NM-NIZK: a useful tool

- Implementation will use *non-malleable non-interactive zero-knowledge proofs* [S99]
- Assume a fixed language $L \in \mathcal{NP}$
- Exists a long random string σ
- "Prover" knows $x \in L$, witness
- Produces a "proof" π of x relative to σ
- "Verifier" checks proof against σ

NM-NIZK: a useful tool (2)

Require four properties

1. (Completeness) If prover has $x \in L$ and witness, then verifier accepts π.
2. (Soundness) If $x \notin L$, no (malicious) prover can make verifier accept
3. (Zero-Knowledge) Proof π reveals nothing about witness
4. (Non-Malleability) A proof π for theorem x cannot be changed into a proof π' for a theorem x'

Sahai's Encryption Scheme

Will build upon previous scheme [S99]

- Recipient key contains:
  - Public portions of *two* (semantically-secure) key pairs
  - Long random string σ
- Sender encrypts m by:
  - Encrypting m in each key
  - Proving (relative to σ) that ciphertexts contain same plaintext
    - "plaintext consistency"
- Receiver decrypts by
  - Checking proof against σ, and
  - If valid, decrypting either ciphertext
- Shown to be secure against chosen-ciphertext attack

ZK proofs of knowledge

We also need proof of *knowledge* [BG92]

- Slight variant to NIZK
- Typically interactive, but still zero-knowledge
- Proves both $x \in L$ *and* prover knows witness
  - Exists extractor that can produce witness
  - Additional information: access to prover

The HLM Scheme

- Receiver key generated as in Sahai's scheme
- Sender public key contains
  1. Semantically-secure encryption key
  2. Signature key
- Sender proves knowledge (in ZK) of decryption key to RA
- RA issues certificate binding together encryption, signature keys
- To encrypt m, sender:
  - Encrypt in all *three* keys
    - Receiver's two and *his own*
  - Prove plaintext consistency
  - Signs ciphertexts, proof
- Decryption as before, plus signature verification

Proofs of security

- Theorem: The HLM scheme is plaintext-aware via key-registration
- Chosen-ciphertext security follows from Sahai's proof
  - Need slightly stronger non-malleability properties of the NIZK proof system
- Plaintext awareness: adversary tries to create a new ciphertext relative to key registered by X
- Two cases:
  - X is honest: extractor simply outputs $\perp$
  - Any other result from D means adversary forged signature

Proofs of security (2)

- X corrupted (key registered by adversary):
  - Registration requires proof of knowledge for secret key
  - Ciphertext contains proof of plaintext consistency
  - Extractor
    - Runs extractor from proof of knowledge system, gets private key
    - Decrypts component ciphertext
  - Technicality: adversary may create “new” ciphertext by changing signature on externally-generated ciphertext
    - Modify definition of “new” ciphertext, or
    - Use scheme with unique signatures (specific assumptions)

Conclusion

- Proposed a new definition of plaintext-awareness
- Uses a more natural model of public-key cryptography
  - Utilizes existing third parties
  - Grants them least possible trust
  - No new network overhead
- Implemented under general assumptions

Future work

- Efficiency
  - General-purpose proof systems inefficient
  - Replace with faster (specific) implementations
- Anonymity
  - Blind sender key?